Gem #152: Определение нового языка программирования в файле проекта

Автор: Vincent Celier - AdaCore

Краткое содержание: Возможно использовать язык программирования, изначально неизвестный `gprbuild`. Этого можно добиться, например, определив все характеристики необходимого языка в файле проекта.

Давайте начнём…

`Gprbuild`, многоязычный конструктор, обладает знаниями о множестве наборов инструментов для разных языков, таких как **Ada**, **C**, **C ++**, **Fortran** и **Assembler**. Эти знания содержатся в файле проекта конфигурации, который обычно создается «на лету» `gprbuild`. Файл проекта конфигурации, который `gprbuild` создает и затем использует, является файлом проекта, который содержит характеристики языков, используемых в дереве пользовательских проектов. Атрибуты, которые определяют эти характеристики, наследуются всеми файлами проекта пользователя.

Однако в некоторых случаях `gprbuild` не знает язык, потому что это другой набор инструментов или другой язык. В этом случае можно объявить все характеристики языка в файле проекта.

Существует много атрибутов конфигурации, которые определяют характеристики языка программирования. Полный список атрибутов можно найти в Руководстве пользователя `GPRbuild` в разделе 1.9.10 (Атрибуты), в частности в разделе 1.9.10.6 (Атрибуты компилятора пакетов).

Характеристики языка программирования

Наиболее важные аспекты определения характеристик языка программирования:

- Схема именования: как `gprbuild` находит источники языка (например: исходные тексты программы)
- Драйвер компилятора и требуемые ключи: как `gprbuild` находит компилятор для языка и минимальные параметры, используемые при вызове компилятора
- Зависимости: как `gprbuild` решает, что источник обновлен или нуждается в перекомпиляции
- Поиск по каталогам: как `gprbuild` указывает компилятору список каталогов для поиска импортированных или включенных файлов.

В качестве упрощенного примера рассмотрим вымышленный язык «**New_Lang**». Компилятор называется "`Nlang`". Исходный суффикс "`.nlng`". Каталоги для поиска указываются с помощью ключа "`-I`". Для создания фрагмента `Makefile`, который содержит зависимости, есть ключ "`--dependencies =`".

Предположим, что язык "**New_Lang**" основан на файлах, а не на единицах. По правде говоря, Ada - единственный основанный на модуле язык, и он требует специальной обработки от `gprbuild`. Все остальные языки, такие как C, C ++ и Fortran, основаны на файлах.

Схема именования

Нам нужно указать `gprbuild`, как найти источники языка **New_Lang**. Как обычно для файловых языков, нам нужно только указать суффикс по умолчанию источников.

В именовании пакетов исходный суффикс указывается с атрибутом `Body_Suffix` (или эквивалентной `Implementation_Suffix`):

```
package Naming is
   for Body_Suffix ("New_Lang") use ".nlng";
end Naming;
```

Драйвер компилятора и необходимые переключатели

Драйвер компилятора определяется с атрибутом `Driver` в пакете `Compiler`:

```
for Driver ("New_Lang") use "nlang";
```

Здесь мы предполагаем, что исполняемый файл "`nlang`" находится в `path`. Если это не так, то мы указываем драйвер компилятора с его полным именем (`path`):

```
for Driver ("New_Lang") use "/path/to/bin/nlang";
```

«Требуемые ключи» (required switches) - это параметры, которые необходимо указать при вызове компилятора, чтобы он правильно выполнял компилирование и только компилировал.

Существует два вида обязательных переключателей: leading и trailing (ведущие и последующие).

Ведущие обязательные переключатели являются первыми переключателями в вызове компилятора и определяются с помощью атрибута `Leading_Required_Switches` (или эквивалентного `Required_Switches`). Конечные обязательные ключи являются последними в вызове компилятора и определяются атрибутом `Trailing_Required_Switches`.

Язык "**New_Lang**" имеет только один единственный обязательный переключатель - это "`-c`":

```
for Leading_Required_Switches ("New_Lang") use ("-c");
```

Зависимости

Существует два вида зависимостей для файловых языков: «`Makefile`» и «`None`». «`None`» - это значение по умолчанию.

Когда тип зависимости "`None`", `gprbuild` будет перекомпилировать исходный код, только если исходный файл был изменен после его последней компиляции. Другими словами, только если отметка времени объектного файла раньше, чем отметка времени источника.

Когда тип зависимости "`Makefile`", `gprbuild` ожидает найти файл зависимости в каталоге объектов. Имя файла этого файла зависимостей получено из имени исходного файла с расширением «`.d`». Например, файл зависимости для источника **C** «`toto.c`» называется «`toto.d`».

Этот файл зависимостей содержит фрагмент `Makefile` для перечисления всех файлов, которые должны быть проверены `gprbuild` при принятии решения о необходимости перекомпиляции источника. Если объектный файл имеет метку времени раньше, чем метка времени любого из этих файлов, то `gprbuild` перекомпилирует исходный код.

Вот содержимое файла зависимостей `toto.d` для источника **C** `toto.c`:

```
toto.o: /path/to/project_dir/sources/toto.c \
 /path/to/project_dir/templates/toto.h
```

Язык "**New_Lang**" имеет вид зависимостей "Makefile". Это означает, что для источника "file_name.nlng" gprbuild должен найти в каталоге объектов фрагмент Makefile "file_name.d". Если какой-либо из файлов, перечисленных в файле зависимостей, является более поздним, чем объектный файл, то gprbuild перекомпилирует исходный код.

Вид зависимости определяется атрибутом Dependency_Kind:

```
for Dependency_Kind ("New_Lang") use "Makefile";
```

Переключатель для создания файла зависимостей при вызове компилятора для нашего вымышленного языка «**New_Lang**» - «--dependencies = .d». Атрибут Dependency_Switches:

```
for Dependency_Switches ("New_Lang") use ("--dependencies=");
```

Поиск в каталогах

Существует несколько способов указать компилятору список каталогов для поиска файлов, необходимых для компиляции источника языка на основе файлов. Например, **C** (gcc) использует переменную окружения "CPATH". Это определяется:

```
for Include_Path ("C") use "CPATH";
```

Наш язык "New_Lang" использует переключатель для каждого каталога, который нужно искать:

```
for Include_Switches ("New_Lang") use ("-I", "");
```

Резюме

Таким образом, возможный файл проекта для нашего языка "**New_Lang**" следующий:

```
project Prj is
   for Languages use ("New_Lang");
   for Source_Dirs use (".", "src1", "src2");
   for Object_Dir use "obj";

   package Naming is
      for Body_Suffix ("New_Lang") use ".nlng";
   end Naming;

   package Compiler is
      for Driver ("New_Lang") use "nlang";
      for Leading_Required_Switches ("New_Lang") use ("-c");
      for Dependency_Kind ("New_Lang") use "Makefile";
      for Dependency_Switches ("New_Lang") use ("--dependencies=");
      for Include_Switches ("New_Lang") use ("-I", "");
   end Compiler;
end Prj;
```

Если каталог проекта "/proj_dir" содержит единственный исходный файл "toto.nlng", а подкаталоги "src1" и "src2" не содержат файла с расширением ".nlng", то используйте команду:

```
gprbuild -v prj.gpr
```

должен привести к единственному вызову компилятора:

```
/path/to/bin/nlang -c --dependencies=toto.d -I /proj_dir -I /proj_dir/src1 -I /proj_dir/src2 /project_dir/toto.nlng
```

и следующие файлы должны быть созданы в объектной директории "obj":

- auto.cgpr (автоматически созданный файл проекта конфигурации)
- toto.o (объектный файл)
- toto.d (файл зависимостей)

Конечно, есть много других атрибутов, которые определяют характеристики языка программирования. Если вы хотите определить новый язык программирования, мы рекомендуем вам изучить все атрибуты конфигурации в Руководстве пользователя GPRbuild.

Специальные языки

Для некоторых языков «объектные файлы» не связаны с исполняемым файлом. Например, это касается **Java**. Для этих языков это указывается атрибутом Objects_Linked на уровне проекта (нет ни в одном пакете в файле проекта):

```
for Objects_Linked ("<language name>") use "False";
```

«Компилятор» для некоторых «языков» может даже не создавать объектный файл. На это указывает атрибут Object_Generated, также на уровне проекта:

```
for Object_Generated ("<language name>") use "False";
```

Для таких языков компилятор вызывается каждый раз, когда вызывается gprbuild.

Есть даже более странные «языки»: языки без компилятора. На это указывает драйвер компилятора, указанный как пустая строка. Одним из примеров является язык «файл проекта», который по умолчанию известен gprbuild.

```
package Compiler is
   for Driver ("project file") use "";
   ...
end Compiler;
```

Такие языки, как «файл проекта» ("project file"), используются GPS. «Источники» ("sources") перечислены в представлении проекта Project View, но, поскольку нет драйвера компилятора, gprbuild не выполняет «компиляцию» ("compilation").

Еще не все...

Если вы широко используете новый язык, вам нужно, чтобы его характеристики автоматически определялись в файле проекта конфигурации, без необходимости помещать все атрибуты конфигурации в каждый из файлов проекта, которые содержат источник вашего языка. Это будет предметом более позднего Gem.

Связанный со статьёй текст программы

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Автор: Vincent Celier - AdaCore

Винсент Селье провел двадцать лет во французском флоте в качестве офицера по радарам и компьютерного оборудования. Он ушел в отставку в 1988 году в ранге командира и присоединился к CR2A, фирме по разработке программного обеспечения, где он был одним из авторов Технического отчета ISO ExtrA (Extensions Temps Reel en Ada). В 1994 году он эмигрировал в Ванкувер, Канада, чтобы работать над CAATS, Канадской автоматизированной системой воздушного движения, большой системой, написанной на языке Ада. Он присоединился к AdaCore в 2000 году. Он является основным разработчиком менеджера проектов и `gprbuild`, с многоязычной поддержкой разработчика.

Last Updated: 10/13/2017
Posted on: 9/23/2013

Обсуждение…